



1. Tibble



개선된 형태의 데이터 프레임

1. Tibble의 생성

- tidyverse의 모든 패키지의 설치: `install.packages("tidyverse")`
- 패키지 tibble의 로딩: `library(tidyverse)`

```
> library(tidyverse)
-- Attaching packages ----- tidyverse 1.2.1 --
✓ ggplot2 3.0.0      ✓ purrr    0.2.5
✓ tibble  1.4.2      ✓ dplyr    0.7.6
✓ tidyr   0.8.1      ✓ stringr  1.3.1
✓ readr   1.1.1      ✓ forcats  0.3.0
-- Conflicts -----tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

- Core tidyverse 패키지가 모두 세션에 올라옴
- Core tidyverse에 속하지 않은 패키지는 따로 함수 `library()`로 불러와야 함

● Tibble의 예

- 패키지 ggplot2의 mpg

```
> mpg
# A tibble: 234 x 11
  manufacturer model    displ  year   cyl trans      drv      cty   hwy fl
  <chr>         <chr>    <dbl> <int> <int> <chr>    <chr> <int> <int> <chr>
1 audi         a4         1.8  1999     4 auto(l~ f      18    29 p
2 audi         a4         1.8  1999     4 manual~ f      21    29 p
3 audi         a4         2    2008     4 manual~ f      20    31 p
4 audi         a4         2    2008     4 auto(a~ f      21    30 p
5 audi         a4         2.8  1999     6 auto(l~ f      16    26 p
6 audi         a4         2.8  1999     6 manual~ f      18    26 p
7 audi         a4         3.1  2008     6 auto(a~ f      18    27 p
8 audi         a4 qua~    1.8  1999     4 manual~ 4      18    26 p
9 audi         a4 qua~    1.8  1999     4 auto(l~ 4      16    25 p
10 audi        a4 qua~    2    2008     4 manual~ 4      20    28 p
# ... with 224 more rows, and 1 more variable: class <chr>
```

- 전통적인 데이터 프레임의 출력 형태와 많이 다른 모습
- Tidyverse에 속한 패키지가 공통적으로 사용하는 개선된 데이터 프레임

- 기존의 데이터 프레임을 tibble로 전환

- 함수 `as_tibble()`

```
> as_tibble(cars)
# A tibble: 50 x 2
   speed  dist
  <dbl> <dbl>
1      4      2
2      4     10
3      7      4
4      7     22
5      8     16
6      9     10
7     10     18
8     10     26
9     10     34
10     11     17
# ... with 40 more rows
```

- 개별 벡터를 이용한 tibble의 생성

- 함수 tibble()

```
> tibble(x=1:3,y=x+1,z=1)
# A tibble: 3 x 3
      x     y     z
  <int> <dbl> <dbl>
1     1     2     1
2     2     3     1
3     3     4     1
```

- 길이가 1인 스칼라만 순환법칙 적용 (만일 z=1:2가 입력되면?)
- 함께 입력되는 변수를 이용한 다른 변수의 생성 가능
- 열(변수) 단위로 입력

- 함수 tibble() vs data.frame()

- 함께 입력된 변수를 이용하여 다른 변수를 만드는 기능

```
> data.frame(x=1:3,y=x+1)
Error in data.frame(x = 1:3, y = x + 1) : object 'x'
not found
```

- 함수 data.frame()에서는 가능하지 않음

- 문자형 벡터를 요인으로 전환하지 않음

```
> tibble(x=1:3, y=letters[1:3])  
# A tibble: 3 x 2  
      x y  
  <int> <chr>  
1     1 a  
2     2 b  
3     3 c
```

```
> str(data.frame(x=1:3, y=letters[1:3]))  
'data.frame':  3 obs. of  2 variables:  
 $ x: int  1 2 3  
 $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

- 행 단위로 입력하여 tibble 생성

- 함수 tribble()

```
> tribble(
  ~x, ~y,
  1, "a",
  2, "b",
  3, "c"
)
# A tibble: 3 x 2
      x y
<dbl> <chr>
1     1 a
2     2 b
3     3 c
```

- 첫 줄: 변수 이름은 '~'로 시작

- 각 자료는 콤마로 구분

2. Tibble과 전통적 데이터 프레임의 비교

- 주된 차이점
 - 데이터 프레임의 출력 방식
 - 인덱싱 방법
 - Row names를 다루는 방식

● 출력 방식의 차이

- 전통적 데이터 프레임: 가능한 모든 자료를 화면에 출력. 대규모 자료의 경우 내용 확인에 어려움

```
> MASS::Cars93
```

- Tibble: 처음 10개 케이스만 출력. 화면의 크기에 따라 출력되는 변수의 개수 조절. 한 화면에서 자료의 특성 파악 용이

```
> as_tibble(MASS::Cars93)
```

```
# A tibble: 93 x 27
  Manufacturer Model    Type Min.Price Price Max.Price MPG.city
*   <fct>         <fct>   <fct>   <dbl>  <dbl>   <dbl>   <int>
1 Acura          Integra Small    12.9   15.9    18.8     25
2 Acura          Legend  Midsi~  29.2   33.9    38.7     18
3 Audi           90      Compa~  25.9   29.1    32.3     20
4 Audi           100     Midsi~  30.8   37.7    44.6     19
5 BMW            535i     Midsi~  23.7   30      36.2     22
6 Buick          Century Midsi~  14.2   15.7    17.3     22
7 Buick          Lesabre Large   19.9   20.8    21.7     19
8 Buick          Roadmas~ Large   22.6   23.7    24.9     16
9 Buick          Riviera Midsi~  26.3   26.3    26.3     19
10 Cadillac      Deville Large    33     34.7    36.3     16
# ... with 83 more rows, and 20 more variables: MPG.highway <int>,
#   AirBags <fct>, DriveTrain <fct>, cylinders <fct>,
#   EngineSize <dbl>, Horsepower <int>, RPM <int>,
#   Rev.per.mile <int>, Man.trans.avail <fct>,
#   Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   wheelbase <int>, width <int>, Turn.circle <int>,
#   Rear.seat.room <dbl>, Luggage.room <int>, weight <int>,
#   Origin <fct>, Make <fct>
```

- 변수 이름과 더불어 변수의 유형을 함께 표시

- 더 많은 자료 확인

```
print(tbl, n=20, width=Inf)
```

- Row names 처리 방식의 차이

- 전통적 데이터 프레임: 자료 출력 시 row name 함께 출력
- Tibble: 생략

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
> mtcars_t <- as_tibble(mtcars)
> print(mtcars_t, n=6)
```

```
# A tibble: 32 x 11
  mpg     cyl  disp    hp  drat    wt  qsec    vs  am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21         6  160   110  3.9    2.62  16.5     0    1     4     4
2  21         6  160   110  3.9    2.88  17.0     0    1     4     4
3  22.8        4  108    93  3.85   2.32  18.6     1    1     4     1
4  21.4         6  258   110  3.08   3.22  19.4     1    0     3     1
5  18.7         8  360   175  3.15   3.44  17.0     0    0     3     2
6  18.1         6  225   105  2.76   3.46  20.2     1    0     3     1
# ... with 26 more rows
```

- 생략된 row name: 변수로 전환

- 함수 rownames_to_column()

```
> mtcars_t <- rownames_to_column(mtcars_t, var="rowname")  
> mtcars_t
```

```
# A tibble: 32 x 12  
  rowname      mpg  cyl  disp  hp  drat    wt  qsec    vs    am  
  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Mazda RX4      21     6  160   110  3.9   2.62  16.5     0     1  
2 Mazda RX4 W~   21     6  160   110  3.9   2.88  17.0     0     1  
3 Datsun 710     22.8    4  108    93  3.85   2.32  18.6     1     1  
4 Hornet 4 Dr~   21.4    6  258   110  3.08   3.22  19.4     1     0  
5 Hornet Spor~   18.7    8  360   175  3.15   3.44  17.0     0     0  
6 Valiant        18.1    6  225   105  2.76   3.46  20.2     1     0  
7 Duster 360     14.3    8  360   245  3.21   3.57  15.8     0     0  
8 Merc 240D      24.4    4  147.    62  3.69   3.19   20      1     0  
9 Merc 230       22.8    4  141.    95  3.92   3.15  22.9     1     0  
10 Merc 280      19.2    6  168.   123  3.92   3.44  18.3     1     0  
# ... with 22 more rows, and 2 more variables: gear <dbl>, carb <dbl>
```

- 전통적 데이터 프레임의 인덱싱 1
 - 열(변수) 선택.
 - `df[[a]]` 또는 `df[a]`의 형식: 벡터 `a`는 숫자형 혹은 문자형
 - `df[[a]]` : 한 변수의 선택. 결과는 벡터
 - `df[a]` : 하나 또는 그 이상의 변수 선택. 결과는 데이터 프레임

<pre>> df2 x y 1 2 a 2 4 b 3 6 c > df2[1] x 1 2 2 4 3 6 > df2[[1]] [1] 2 4 6</pre>	<pre>> df2["x"] x 1 2 2 4 3 6 > df2[["x"]] [1] 2 4 6</pre>
--	---

- 데이터 프레임의 변수 선택
 - 벡터 형태로 선택하는 것이 일반적
 - `df[[a]]`의 형태가 더 많이 사용됨
 - 조금 더 편한 방법: `$` 기호 사용

```
> df2[["x"]]  
[1] 2 4 6
```

```
> df2$x  
[1] 2 4 6
```

- 전통적 데이터 프레임의 인덱싱 2

- 행렬의 인덱싱 방법 사용
- `df[i,j]`의 형태
- 선택된 변수가 하나이면 결과는 벡터
하나 이상이면 결과는 데이터 프레임

```
> df2[c(1,2),1]  
[1] 2 4
```

```
> df2[c(1,2),]  
  x y  
1 2 a  
2 4 b
```

- 인덱싱 방법의 차이

- 기호 '\$'을 이용하는 경우: 변수 이름의 부분 매칭 허용 여부

전통적 데이터 프레임: 부분 매칭 허용

```
> df1 <- data.frame(xyz=1:3, abc=letters[1:3])  
> df1$x  
[1] 1 2 3
```

tibble: 부분 매칭 불허

```
> tb1 <- as_tibble(df1)  
  
> tb1$x  
NULL  
Warning message:  
Unknown or uninitialised column: 'x'.  
  
> tb1$xyz  
[1] 1 2 3
```

● 연습문제

- 다음의 tibble를 생성하라. 단, 변수 w는 알파벳 소문자 중 임의로 9글자를 선택하여 생성된 것이므로 다른 문자가 선택될 수 있음.

```
> df1
# A tibble: 9 x 4
      x     y     z     w
  <dbl> <int> <int> <chr>
1     1     1     1     q
2     1     2     1     x
3     1     3     1     i
4     1     4     2     h
5     1     5     2     v
6     1     6     2     o
7     1     7     3     g
8     1     8     3     u
9     1     9     3     b
```


- df1의 두 번째 열 선택하여 벡터로 출력하는 세 가지 인덱싱 방법은?
- df1의 두 번째 열의 처음 다섯 개 자료를 행렬의 인덱싱 방법으로 선택
- df1을 전통적인 데이터 프레임으로 전환하고 두 번째 열의 처음 다섯 개 자료 선택. tibble을 대상으로 했을 때와의 차이점은?